A Simulation Scheduling Module to Improve User Experience in the Simugan Beef-Cattle Farm Simulator

M. Arroqui, F. Ferreira, M. Roselli, J. Rodriguez Alvarez, C. Machado, C. Mateos and A. Zunino

***Abstract*—At the Faculty of Veterinary Sciences of the National University of Central Buenos Aires a client-server Beef-Cattle Farm simulator called Simugan has been developed. Simugan allows users to experiment over a virtual farm in a simple and low cost way compared with real farm conditions. Users can submit single simulation scenarios or multiple simulation scenarios packaged in an *experimentation*, where each scenario is a complete farm configuration. This is a key feature important in farm research, but with the drawback that some users might experiment long wait times for simulation results because of the amount of simulations the underlying hardware architecture has to process. Consequently, an heuristic scheduler module was added to Simugan producing a more equitative use of computer resources and an improvement of 41 % in users flow time, a popular metric to quantify how much time user simulations spend in the back end and hence a way of measuring deviations in user's waiting times.**

***Index Terms*—Scheduling, Simugan, Distributed Computing**

I. Introducción

A partir del aumento de la población mundial, se requerirá un 60 % más de alimento para el año 2050 [1]. La ganadería aporta proteína de calidad, sin embargo hay un cuestionamiento ambiental creciente por su contribución con gases de efecto invernadero al fenómeno de Cambio Climático, lo que demanda un análisis integral de los diferentes sistemas productivos. El sector agroindustrial es uno de los pilares económicos de Argentina y con una ganadería vacuna que posee más 54 millones de cabezas [2]. Es de destacar que el crecimiento de las últimas décadas del área total cultivada en el país, fundamentalmente de soja, desplazaron a la ganadería a suelos de menor calidad [3], lo que causó la coexistencia de una diversidad de sistemas productivos ganaderos en las diferentes regiones.

A nivel global, los sistemas ganaderos comparten la diversidad, la interacción importante de factores productivos, económicos y ambientales, que causan dificultad de aislarlos en investigación a campo por lo que es cada vez más creciente su incorporación a herramientas de simulación matemática [4]–[6]. A nivel nacional, a los fines de capturar distintas características de la región pampeana Argentina, se desarrolló un simulador Web de alternativas agropecuarias llamado Simugan [7]. Simugan es un simulador diario ganadero Web desarrollado en la Facultad de Ciencias Veterinarias de la UNICEN mediante la utilización de métodos ágiles [8] que ha sido exitosamente utilizado, entre otros usos, para explorar oportunidades de innovación sustentable [9]. El simulador contiene modelos bioinformáticos que interactúan para representar la biología de base. Además, el simulador posee reglas de manejo empresarial para determinar distintas alternativas a aplicar sobre el establecimiento agropecuario. Este simulador arroja como resultados variables productivas, económicas y ambientales. Simugan es utilizado para realizar simulaciones complejas con varias repeticiones (representando cada una variaciones de clima) dentro del ámbito académico [7], [10]–[13].

Desde el punto de vista del software, Simugan es un sistema cliente-servidor (http://simugan.vet.unicen.edu.ar/simfarm/) que puede ser utilizado de dos modos. En el primer modo un usuario crea un *escenario de simulación*, para luego enviar a ejecutar la simulación asociada y esperar por los resultados de la misma, los cuales deberían estar disponibles en cuestión de unos pocos minutos. En general este tipo de usuario busca generar rápidamente un escenario de simulación base sobre el cual luego realizará una *experimentación* que represente sus intereses de investigación disciplinar. El segundo modo permite a un usuario realizar una experimentación, esto es, una vez alcanzado el escenario base aplica variabilidad sobre determinadas variables, generando así numerosas simulaciones a ejecutarse. Por ejemplo, si se parte de un escenario base donde se detallan las características iniciales de un campo a simular dentro de las cuales se encuentra la cantidad inicial de animales y el clima a utilizar, el usuario podría querer agregar variabilidad a éstas, esto es, simular ese escenario base con distinta cantidad de animales (N) y distintos climas (M). Esto a su vez generaría $N * M$ escenarios a simular y por lo tanto a ejecutar en el servidor.

La ejecución de múltiples experimentaciones de forma concurrente, por parte de uno o varios usuarios utilizando Simugan en el primer modo actualmente genera tiempos de respuesta inadecuados tanto para los usuarios utilizando Simugan en el primer modo como en el segundo (en adelante, al primer modo lo llamaremos *modo simulación* y al segundo *modo experimentación*). Por consiguiente, este trabajo tiene por objetivo describir la mejora en los tiempos de respuestas obtenidos para cada uno de los perfiles de usuarios mencionados mediante la aplicación de un algoritmo de planificación de simulaciones heurístico, que se propone en este artículo.

El resto del artículo está organizado de la siguiente manera: la Sección II describe planificadores de tareas y algoritmos de

National University of Central Buenos Aires

M. Arroqui, C. Mateos and A. Zunino are funded by Consejo Nacional de Investigaciones Científicas y Técnicas de Argentina (CONICET)

planificación existentes. Adicionalmente, ofrece una breve descripción del simulador agropecuario Simugan. La Sección III describe la implementación de un algoritmo de planificación específicamente para Simugan, que apunta a mejorar simultáneamente los tiempos de ejecución de simulaciones bajo ambos modos de operación. La Sección IV presenta un contexto de experimentación para el algoritmo implementado y los resultados obtenidos. Finalmente, la Sección V presenta las conclusiones de este trabajo.

II. Materiales y Métodos

A. Planificación de Tareas

En el contexto de sistemas paralelo/distribuidos, la planificación de tareas se encarga de la asignación de recursos computacionales a los cómputos de los usuarios durante el transcurso en el tiempo. Entonces, un problema de planificación se presenta en la forma de "*qué recurso(s) le asigno a qué usuario -y por consiguiente a sus tareas- y durante cuánto tiempo*". Mayormente este problema tiene tres componentes [14]: (i) la carga de trabajo, que define a los consumidores de recursos en término de las tareas a ejecutar; (ii) los recursos necesarios para ejecutar la carga de trabajo, consistente de un conjunto de nodos o computadoras distribuidas, con uno o más núcleos de procesamiento, conectados por una red, generalmente de alta velocidad. Estos recursos pueden organizarse en un entorno de red local (por ejemplo centro de datos en una Universidad) o en infraestructuras distribuidas y más escalables (Clouds o Grids); (iii) los requisitos de planificación, que determinan el objetivo de la planificación, y otros requisitos que debe cumplir la solución. Por lo general, el objetivo de la planificación es optimizar una o una combinación de métricas de rendimiento tales como el uso de CPU (del inglés Central Processing Unit), el tiempo de espera, el tiempo de flujo o el rendimiento.

La planificación de tareas es una actividad dinámica donde la carga de trabajo y la disponibilidad de los recursos organizados varían en el tiempo, y donde el propósito es eficientizar los objetivos de la planificación.

B. Algoritmos de Planificación

Un planificador de tareas puede considerarse como un componente de software encargado de la asignación de tareas a recursos, que adopta un criterio de asignación concreto y se diseña para un ambiente de procesamiento específico [15]. Este componente tiene como objetivo principal optimizar el uso de los recursos cumpliendo con un conjunto finito de requisitos que le sean relevantes a los usuarios del sistema que se quiere mejorar.

Para cumplir su propósito, el planificador implementa pragmáticamente un algoritmo de planificación. Un algoritmo de planificación puede ser clasificado como heurístico, metaheuristico o evolutivo. Un algoritmo heurístico es aquel que se limita a proporcionar una buena solución al problema, que puede no ser la óptima, pero en tiempos computacionales razonables [16].

En esta línea, existen trabajos en el área de software para simulación agroganadera que explotan técnicas de planificación o en su defecto plataformas para computación distribuida, a fin de acelerar simulaciones costosas computacionalmente que lidian con tiempos de CPU elevados o bien gran cantidad de datos agroganaderos a procesar.

CYBELE [17] es una plataforma que tiene como usuario destino los diversos actores de la cadena de producción y comercialización de productos alimenticios agrícolas, y apunta específicamente a proveer funcionalidad para el acceso integrado y procesamiento de diversos tipos de data-sets, con el fin de generar valor agregado a partir de los data-sets e introducir mejoras en los procesos productivos (predicción de niveles adecuados de fertilizantes y funguicidas, monitoreo de bienestar animal, etc). CYBELE enfatiza la integración con infraestructuras HPC, pero no prescribe algoritmos propios de planificación más allá de los soportados por frameworks base considerados por la plataforma como Apache Spark y Apache Hadoop.

Asimismo, en [18] se describe un framework abierto de servicios de geoprocesamiento en nube aplicados a resolver desafíos agroganaderos mediante simulación y HPC. Este trabajo a su vez abarca tres subproyectos: EUXDAT (EUro-pean e-infrastructure for eXtreme Data Analytics in sustainable developmenT), CYBELE [17], y EOPEN (opEn interOperable Platform for unified access and analysis of Earth observatioN data), para ofrecer acceso a cómputo de alto rendimiento en nube, soporte para Big Data, y acceso amigable para acceso y visualización de resultados de simulación. Para el procesamiento en nube, el framework utiliza MPI (Message Passing Interface), un estándar clásico para implementación de sistemas paralelo/distribuidos mediante el paradigma de pasaje de mensajes.

Por otra parte, el framework propuesto en [19] apunta a la maximización (a escala de rodeo) del área sembrable o utilizable para pastoreo mediante la simulación de modelos hidrológicos complejos. Específicamente, el trabajo adopta el modelo APEX – Agricultural Policy/Environmental eXtender Model (https://epicapex.tamu.edu/apex/) y propone un soporte para paralelizar dichos modelos, explorando técnicas conocidas de computación de alto desempeño, lo que permite escalar simulaciones espaciales al tiempo de reducir los tiempos de ejecución de los modelos.

Lo que se desprende de los trabajos relacionados citados es que para administrar las tareas producto de la paralelización del procesamiento de datos ([17], [19]) o de los modelos ([19], [18]), los trabajos toman prestado técnicas/planificadores ya existentes en el área de Computación Paralela y Distribuida. Por el contrario, para el simulador objetivo en este trabajo, esto es Simugan, en este paper se propone un nuevo planificador de dominio específico, que especializa planificadores de propósito general teniendo en cuenta las particularidades de los tipos de cómputos de Simugan y los tipos de usuarios que intervienen en el proceso de simulación prescrito en el uso típico del simulador.

C. Simulador Agropecuario: Simugan

Simugan [7] es un simulador agropecuario desarrollado con el propósito de dar soporte a la toma de decisiones en

ganadería de carne vacuna. Este simulador permite modelar las características de un campo y observar las consecuencias de distintos cursos de acción, permitiendo al usuario aprender de los mismos de una forma eficiente y económica, siendo un soporte valioso para la toma de decisiones. Actualmente, su desarrollo ha implicado 12500 horas/hombre de trabajo y debido al uso recurrente y complejidad de cálculo es que se ejecuta sobre una Grid Computacional [20]. Adicionalmente, cuenta con capacidad de procesamiento de grandes volúmenes de información [21], así como también acceso desde clientes móviles [22].

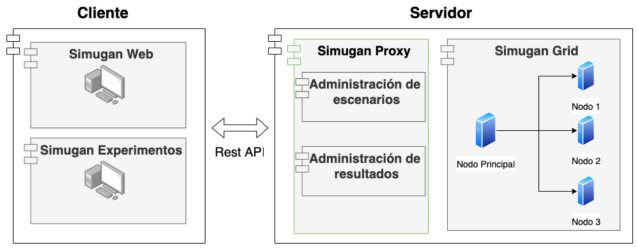


Fig. 1. Arquitectura de Simugan.

En la Figura 1 se muestra un diagrama simplificado de la arquitectura de Simugan. En el lado cliente, se cuenta con dos aplicaciones de interfaz bien definidas, una de ellas – Simugan Web implementada con OpenLazlo (https://www.openlaszlo.org/) y en proceso de migración a ReactJS (https://es.reactjs.org/) – para la configuración del modo simulación y la otra – Simugan Experimentos implementada con ReactJS -- para la configuración de experimentos basados en un escenario de simulación ya creado, esto es, modo experimentación. En el lado servidor, implementado utilizado Spring (https://spring.io/), se encuentra un proxy que se encarga de la administración e inyección de escenarios a la Grid Computacional y de la administración de los resultados de simulación. Finalmente, la Grid Computacional se ha montado utilizando el middleware Apache Ignite (https://ignite.apache.org/).

Por otro lado, un escenario de simulación es la representación completa de un campo, en el cual se configuran aspectos relevantes tales como la cantidad de potreros que tiene el campo, la rotación de cultivos y pasturas que tiene ese potrero, cuántos rodeos de animales hay, cantidad de animales que tiene el rodeo, características biofísicas de los animales (por ejemplo edad inicial, peso inicial, condición corporal inicial, raza, entre otras), etc. Además, se debe especificar la fecha de inicio de la simulación y la fecha de fin, para las cuales se puede utilizar algunos de los climas pre-cargados de Simugan o utilizar un clima ad-hoc creado por el usuario. Cabe destacar que el simulador simula animal a animal con un paso de tiempo diario.

III. Implementación de un Algoritmo de Planificación en Simugan

Simugan en su componente administrador de escenarios (Figura 1) no cuenta con un planificador de simulaciones, por lo que la inyección a la Grid Computacional se realiza por orden de llegada, para luego ejecutar las simulaciones de forma FIFO (del inglés First In, First Out). La falta de un planificador de simulaciones causa esperas demasiado largas en usuarios que se encuentran en modo simulación y un uso desbalanceado de los recursos de la Grid Computacional para todos los usuarios en general, independientemente del modo de operación.

Por ejemplo, si un usuario en modo experimentación, sobre un escenario base aplica 5 estrategias de alimentación de ganado diferentes, 5 cantidades deferentes de animales (carga animal) y 4 variedades climáticas, se obtienen 100 simulaciones a simular, a lo cual hay que agregar el factor de variabilidad estadística (asumamos por ejemplo 10), lo que da un resultado total de 1000 simulaciones a simular. Si estas 1000 simulaciones llegan al servidor en un instante de tiempo t y en el tiempo $t+1$ llega una simulación de un usuario trabajando en modo simulación, esta última simulación es ejecutada *luego* de ejecutar las 1000 simulaciones previas, lo que conlleva a tiempos de espera desiguales. Algo similar sucede con un usuario en modo experimentación, pero que sólo genera 100 simulaciones a simular, el cual debe esperar por las 1000 simulaciones ya presentes en la Grid Computacional. Para solucionar los problemas mencionados se diseñó e implementó un algoritmo de planificación acorde.

A. Algoritmo de Planificación

El algoritmo de planificación considera a un usuario en modo de creación de un escenario de simulación más prioritario que un usuario en modo experimentación ya que los primeros solicitan el procesamiento de pocos escenarios, y requieren tiempos de respuesta bajos para dado el escenario base creado proseguir con el modelado experimental off-line tan pronto como sea posible. De este modo siempre que hubiera escenarios por simularse de usuarios en modo simulación, tendrían prioridad por sobre los escenarios de experimentación para inyectarse a la Grid Computacional. En tanto, para la ponderación de los escenarios pertenecientes a cada grupo se seleccionó por un lado un conjunto de parámetros relevantes para el dominio de simulaciones agropecuarias, y por el otro, parámetros que representan el estado de un escenario a simularse. Cada parámetro se compone de dos valores, un valor de prioridad respecto al resto de los parámetros y un valor en forma de rango de ese parámetro, generando ecuación de ponderación (1), donde P_i representa un valor de prioridad de un parámetro en particular en el rango [1-10], V_i representa un valor dentro del rango designado al parámetro P_i, R es el valor total de ponderación para un escenario en particular, y n en la cantidad de parámetros a utilizar.

$$R = P_1 \cdot V_1 + P_2 \cdot V_2 + P_3 \cdot V_3 + ... + P_n \cdot V_n = \sum_{i=1}^{n} P_i \cdot V_i \quad (1)$$

La selección de qué parámetros se utilizaron, qué prioridad P_i le corresponde a un parámetro en particular y qué valor de rango V_i tendrá este parámetro fue basada en el algoritmo de planificación "Primero el trabajo más corto", esto es, que las simulaciones más cortas sean las primeras en inyectarse a la Grid Computacional [23]. A continuación se

detallan los parámetros utilizados junto con su ponderación y rangos. Cabe destacar que la selección de parámetros, prioridad y rango fue un trabajo conjunto entre los expertos del dominio agropecuario y los ingenieros de sistemas que dan soporte al proyecto Simugan (https://www.researchgate.net/project/Simugan) y de los modelos productivos regionales que pública trimestralmente el Ministerio de Agricultura, Ganadería y Pesca de la Argentina (https://www.magyp.gob.ar/sitio/areas/bovinos/modelos/).

1) Parámetro 1: Cantidad de Escenarios por Simular de un Usuario: Se decidió que un usuario con menor cantidad de escenarios por inyectar, tiene mayor valor de rango que otro con mayor cantidad de escenario por inyectar. El objetivo fue evitar el problema de que un mismo usuario inunde la Grid Computacional durante largos periodos de tiempo. En cuanto a los valores de ponderación, se determinó utilizar cinco rangos para calificar a un usuario con más prioridad que otro. Los valores son: (i) entre 0 y 10 escenarios por simular con valor de 10; (ii) entre 11 y 20 escenarios por simular con valor de 7; (iii) entre 21 y 40 escenarios por simular con valor de 5; (iv) entre 41 y 70 escenarios por simular con valor de 3; (v) más de 71 escenarios por simular con valor de 1. En cuanto a la prioridad global de este parámetro (respecto de R), se determinó como el más prioritario, otorgando un valor de prioridad de 0,33.

2) Parámetro 2: Tiempo de Espera del Escenario: Se decidió que cuanto más tiempo de espera tiene un escenario por inyectarse, mayor sea su valor de rango. De esta forma, se evita el problema de que un escenario sea valorado con una puntuación baja y quede mucho tiempo en espera, y por ende, sufra *inanición*. Los valores son: (i) mas de 190 minutos con un valor de 10; (ii) entre 160 y 189 minutos con un valor de 8; (iii) entre 130 y 159 minutos con valor de 7; (iv) entre 100 y 129 minutos con un valor de 5; (v) entre 70 y 99 minutos con un valor de 3; (vi) entre 40 y 49 minutos con un valor de 2; (vii) entre 0 y 39 minutos con valor de 1. En cuanto a la prioridad global de este parámetro, se determinó como el segundo más prioritario, otorgando un valor de prioridad de 0,27.

3) Parámetro 3: Cantidad de Años a Simular de un Escenario: Se decidió que una cantidad menor de años de simulación en un escenario tiene mayor valor de rango. Este valor es de notable importancia a la hora de ponderar ya que la duración en años de un escenario es directamente proporcional al costo computacional, esto es a mayor cantidad de años de simulación, mayor será la cantidad de procesos a realizar y por lo tanto el tiempo de procesado de la simulación será más extenso. Los valores en este caso son: (i) entre 0 y 10 años de simulación con un valor de 10; (ii) entre 11 y 20 años de simulación con un valor de 7; (iii) entre 21 y 40 años de simulación con un valor de 4; (iv) mas de 41 años de simulación con un valor de 2. En cuanto a la prioridad de este parámetro, se determinó como el tercero más prioritario, otorgando un valor de prioridad de 0,2.

4) Parámetro 4: Cantidad de Rodeos: Se decidió que una cantidad menor de rodeos de animales en un escenario tiene mayor valor de rango. De forma similar al parámetro anterior, la cantidad de rodeos impacta directamente en el costo computacional del escenario a simular. Los valores son: (i) entre 0 y 2 rodeos con valor de 10; (ii) entre 3 y 4 rodeos con valor de 7; (iii) entre 5 y 6 rodeos con valor de 4; (iv) más de 7 con valor de 2. En cuanto a la prioridad global de este parámetro, se determinó como el cuarto más prioritario, otorgando un valore de prioridad de 0,13.

5) Parámetro 5: Cantidad de Animales: Se decidió que una cantidad menor de animales en un escenario tiene mayor valor de rango. Intuitivamente, la cantidad de animales impacta directamente en el costo computacional del escenario a simular. Los valores son: (i) entre 0 y 600 animales con valor de 10; (ii) entre 601 y 1200 animales con valor de 8; (iii) entre 1201 y 1800 animales con valor de 7; (iv) entre 1801 y 2400 animales con valor de 5; (v) entre 2401 y 3000 animales con valor de 3; (vi) entre 3001 y 3600 animales con valor de 2; (vii) más de 3601 animales con valor de 1. En cuanto a la prioridad de este parámetro, se determinó como menos prioritario, otorgando un valor de prioridad de 0,07.

6) Selección del Escenario Prioritario: Cada uno de los grupos de escenarios, los creados por usuarios en modo simulación y los creados en modo experimentación, son ponderados cada cierto tiempo por el algoritmo de mayor a menor valor de R. Siempre que haya escenarios por ejecutar en el primer grupo, el algoritmo seleccionará estos ordenados por R, y cuando no haya más escenarios por inyectarse en el primer grupo, se seleccionaran los del segundo, también ordenados por R de mayor a menor.

B. Implementación del Algoritmo de Planificación

Se agregó un módulo que implementa esta lógica de planificación en el componente Administración de escenarios (Fig. 1) de Simugan, donde para cada escenario de simulación cuando éste arriba al lado servidor se agrega la información extra necesaria para la implementación del algoritmo de planificación. Los información agregada fue respecto a un escenario a simular es: (i) fecha y hora cuando el escenario a ejecutar estuvo disponible para inyectarse a la Grid Computacional; (ii) fecha y hora donde se calculó por última vez su prioridad R; (iii) último valor de su prioridad R; (iv) información de la cantidad de rodeos que tiene el escenario; (v) información de la cantidad de animales que tiene el escenario; (vi) información de la cantidad de años que deberá ejecutarse el escenario a simular. Respecto al usuario, a su vez se registra la siguiente información: (i) cantidad de escenarios experimentales que tiene el usuario en particular; (ii) cantidad de escenarios en modo simulación que tiene el usuario en particular; (iii) cantidad de escenarios en modo experimentación que finalizaron su ejecución; (iv) cantidad de escenarios en modo simulación que finalizaron su ejecución; (v) tiempo de uso de la Grid, esto es, la suma total de tiempos que los escenarios inyectados han tardado en simularse dentro de la Grid; (vi) tiempo de espera, esto es, la suma total de tiempos que los escenarios han esperado para ser inyectados a la Grid; (vii) tiempo total, esto es, la suma total de tiempos que los escenarios arribados a Simugan han tardado en simularse y producir resultados.

La implementación se realizó con tres colas de prioridades, una cola para los escenarios experimentación, otra para los escenarios en modo simulación y la última para los usuarios.

Cada cola de prioridades son instancias únicas, patrón de diseño Singleton [24], por lo que se mantienen de principio a fin durante el tiempo de ejecución. Dichas instancias al momento de iniciar la aplicación son solicitadas al framework Spring. Este se encarga de crearlas y mantenerlas en memoria por el tiempo que la aplicación esté en ejecución.

Cuando un escenario o una experimentación arriba, se lo inserta en la cola correspondiente y se calculan y almacenan la información mencionada para poder ser usada durante las ponderaciones. Adicionalmente, se actualiza la cola de usuarios, agregando un nuevo usuario o si este ya está a la espera de ejecución se actualizan sus datos. El Algoritmo 1 muestra el pseudocódigo, que se ejecuta cada 2 segundos, donde inicialmente se chequea la carga de la Grid , luego se actualiza la ponderación y finalmente se inyecta a la Grid tantos escenarios como la carga de la misma lo permita. La carga de la Grid está definida como la cantidad de simulaciones sobre la cantidad de nodos de la Grid.

A modo de ejemplo, si el Usuario 1 que tiene actualmente 15 escenarios por simularse, envía a simular un escenario con 2 rodeos de 260 animales cada uno, 5 años de simulación, el mismo tendrá un valor $R = 0,33 \cdot 7 + 0,27 \cdot 1 + 0.2 \cdot 10 + 0,13 \cdot 10 + 0,07 \cdot 10 = 6,58$. En espera para ser inyectado a la Grid, desde hace 50 minutos, se encuentra un escenario de Usuario 2, quien tiene adicionalmente 100 escenario más por simular. Este escenario tiene 2 rodeos de 600 animales cada uno y 15 años de simulación, entonces $R = 0,33 \cdot 1 + 0,27 \cdot 3 + 0.2 \cdot 7 + 0,13 \cdot 10 + 0,07 \cdot 8 = 4,4$. De esta forma, el escenario inyectado por el Usuario 1 será ponderado mayor que el ya existente del Usuario 2.

```
void schedulingAlgorithimPseudoCode(){
 if (Grid.getWorkLoad() < 1){
  Queue queue = getQueue();
  calculateRforScenarios(queue);
  do{
    SimulationScenario s = queue.peak();
    Grid.inject(s);
  }while(Grid.getWorkLoad() < 1)
 }
}

double calculateR(Scenario s){
 return calculateP1(s) + calculateP2(s)
        +calculateP3(s) + calculateP4(s)
        + calculateP5(s);
}
```

Algoritmo 1: Pseudocódigo Java del algoritmo de planificación.

IV. Experimentación

A. Ambiente de Ejecución de Pruebas

Para llevar a cabo la experimentación y análisis del planificador, se implementó un entorno Grid simulado por software representando el comportamiento de la Grid Computacional que posee Simugan. Esto permitió hacer pruebas experimentales de mayor magnitud, permitiendo usar una mayor cantidad de nodos de procesamiento a la hora de experimentar, y así evitar contar con el hardware físico. Para ejecutar las simulaciones en sí se utilizó un servidor HP Proliant DL 160 G9 con dos procesadores Intel Xeon E5-2609v4 y 32 GB de memoria RAM.

Por otro lado, se utilizó la versión 8 de Java y la herramienta IDE Eclipse JEE 2020-03 tanto para el desarrollo del entorno Grid como para la obtención y análisis de resultados de las pruebas. Desde el lado del usuario, se emuló su interacción con el portal Simugan Web original a través de una herramienta llamada Postman (https://www.getpostman.com/) para enviar peticiones programáticamente al servidor. El objetivo de utilizar una herramienta como Postman no sólo apunta a consumir servicios REST (del inglés REpresentational State Transfer) creados para la generación y procesamiento de tareas, sino también para crear un conjunto de peticiones HTTP (del inglés HyperText Transfer Protocol) sincronizadas y automáticas que permiten crear un contexto más "realista" al realizar las pruebas, evitando realizar todas las peticiones en el mismo momento, situación poco común sobre el sistema de Simugan.

B. Variantes Experimentales

Se diseñó un conjunto de experimentos que buscaron medir el *flow time* [25] de cada usuario en modo simulación y en modo experimentación, bajo el esquema de ejecución de Simugan actual versus la utilización del planificador propuesto. El flow time Ft hace referencia al período de tiempo requerido para completar una determinada tarea y, como se muestra en (2) se define como la suma de los tiempos transcurridos entre la llegada de cada tarea hasta su finalización, donde TG es el tiempo desde que el escenario llega al servidor, hasta que es insertado en su cola correspondiente para el usuario i, TE es el tiempo de espera que tiene el usuario i en la cola de simulación y/o experimentación, y TP es el tiempo de procesamiento que tiene el usuario i desde que inyecta un escenario de simulación a la Grid Computacional, hasta que es el resultado está disponible.

$$Ft_i = TG_i + TE_i + TP_i \tag{2}$$

Cabe destacar que como se expuso en la sección anterior la prioridad de los escenarios está dado por cinco factores, donde tres de ellos son constantes durante la ejecución (cantidad de años, cantidad de rodeos y cantidad de animales) y las otras dos (cantidad de escenarios por simular de un usuario y tiempo de espera del escenario) son las que contribuyen a elevar o no la valoración del escenario en el orden de la cola. De éstas últimas dos, la más preponderante es a su vez la cantidad de escenarios por simular de un usuario, y por este motivo se decidió experimentar con la puntuación otorgada por el planificador bajo tres diferentes alternativas: 1) Darle más prioridad a usuarios con menor cantidad de escenarios en la cola de espera. 2) Darle más prioridad a usuarios con mayor cantidad de escenarios en la cola de espera. 3) Darle más prioridad a usuarios con menos de 150 escenarios y luego priorizar a los que tengan más cantidad de escenarios. Se consideró el número de 150 para priorizar experimentaciones de menor tamaño. Para cada alternativa, adicionalmente se calculó la desviación estándar de Ft (3), donde FtT es el flow time total para un usuario dado en una Alternativa de

experimentación, n en la cantidad de usuarios, $\overline{FtT}$ es el flow time promedio de la promedio de la alternativa, y N es el tamaño de la muestra. Con el cálculo de la desviación estándar se pudo fijar la cota superior y la cota inferior en la cual pueden oscilar los valores. Dichos límites se calculan en base a las fórmulas $L_s = \overline{FtT} + S$ y $L_i = \overline{FtT} - S$ donde L_s es la desviación máxima y L_i es la desviación mínima.

$$S = \sqrt{\frac{\sum_{i=1}^{n} \left(FtT_i - \overline{FtT}\right)^2}{N-1}} \tag{3}$$

Para analizar el comportamiento del planificador, se realizaron ejecuciones con 10 usuarios en modo experimentación y 6 usuarios en modo simulación ejecutando de forma concurrente, valores seleccionados en base al uso de Simugan. En todos los casos se utilizó un flujo de entrada de usuarios constante, esto es, cada 5 minutos se conecta un nuevo usuario, en pos de simular un uso real de Simugan. Adicionalmente, en el armado de la prueba se buscó que 2 usuarios de experimentación se ubiquen en cada uno de los rangos de cantidad de escenarios para analizar la variación de la puntuación de la prioridad total de los escenarios. La Tabla I muestra la configuración de los usuarios con los cuales se experimentó, donde se muestra la cantidad de escenarios que tiene cada uno, qué tipo de usuario es y el tiempo de demora desde el comienzo del experimento hasta su ingreso.

TABLA I
USUARIOS UTILIZADOS EN LA EXPERIMENTACIÓN.

Id de usuario	Cantidad de escenarios	Tipo de usuario	Tiempo de demora
6	200	Experimentación	0 minutos
12	1	Simulación	5 minutos
4	135	Experimentación	10 minutos
1	50	Experimentación	15 minutos
11	1	Simulación	20 minutos
10	500	Experimentación	25 minutos
2	100	Experimentación	30 minutos
15	1	Simulación	35 minutos
9	700	Experimentación	40 minutos
13	1	Simulación	45 minutos
7	350	Experimentación	50 minutos
14	1	Simulación	55 minutos
3	150	Experimentación	60 minutos
16	1	Simulación	65 minutos
5	250	Experimentación	70 minutos
8	400	Experimentación	75 minutos

C. Resultados Obtenidos

En la Tabla II se muestra el tiempo de entrada, tiempo de salida y flow time de cada uno de los usuarios en modo simulación en minutos sin el planificador propuesto y con el planificador. Respecto al flow time, se aprecia con claridad la diferencia entre los tiempos que tardaron las simulaciones de cada uno de los usuarios. En el caso de la configuración sin planificador los usuarios que ingresan sobre el final, tienen mucho tiempo de espera debido a que Simugan está atendiendo las simulaciones de los usuarios en modo experimentación.

En la Fig. 2 se muestra el tiempo de entrada, tiempo de salida y flow time de cada uno de los usuarios en modo

TABLA II
FLOW TIME DE LOS USUARIOS EN MODO SIMULACIÓN. TE = TIEMPO DE ENTRADA, TF = TIEMPO DE FINALIZACIÓN, FT = FLOW TIME.

Usuario	Sin planificador	Con planificador
U12	Te=5, Tf=36, Ft=31	Te=5, Tf=22, Ft=17
U11	Te=20, Tf=56, Ft=36	Te=20, Tf=33, Ft=13
U15	Te=35, Tf=130, Ft=95	Te=35, Tf=47, Ft=12
U13	Te=45, Tf=220, Ft=175	Te=45, Tf=61, Ft=16
U14	Te=55, Tf=262, Ft=207	Te=55, Tf=67, Ft=12
U16	Te=65, Tf=278, Ft=213	Te=65, Tf=73, Ft=8

experimentación para la Alternativa 1 (Fig. 2a), la Alternativa 2 (Fig. 2b) y la Alternativa 3 (Fig. 2c) de configuración del planificador de simulaciones. Comparando los tres gráficos se puede observar que el Usuario 9 que tiene 700 escenarios para simular, tiene un menor flow time en la Alternativa 3, luego en la Alternativa 1 y por último en la Alternativa 2. El mejor flow time sucede en la Alternativa 3 dado que si en la cola de espera de usuarios experimentación no hay escenarios de usuarios con menos de 150 escenarios, se empieza a priorizar a los usuarios con mayor cantidad de escenarios. De forma similar ocurre en el Usuario 10, hasta que en el minuto 40 arriba el Usuario 9 (Tabla I) y este último pasa a tener mayor prioridad debido a que tiene mayor cantidad de escenarios. Por esto es que el flow time del Usuario 10 es menor en la Alternativa 1. Por su parte el flow time del Usuario 7 tiene un mejor desempeño bajo la Alternativa 1 debido a que en la Alternativa 3, cuando arriba en el minuto 50, el Usuario 9 aún tiene mayor cantidad de escenarios por simular. Por su parte, la Alternativa 2, que prioriza a los usuarios con mayor cantidad de simulaciones, fue la que peor desempeño tuvo debido a que usuarios con poca cantidad de escenarios por simular debían esperar por los usuarios con mayor cantidad, ocasionando grandes tiempos de flujo.

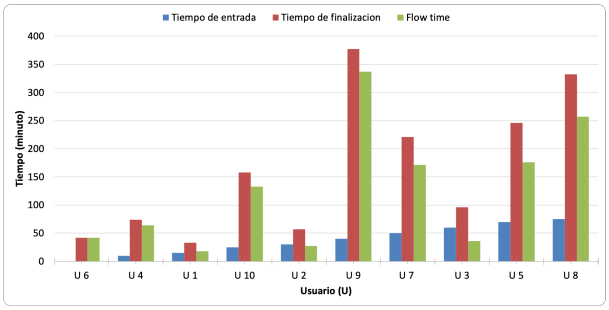
En la primera fila de la Tabla III se observa el flow time total de cada una de las alternativas. La segunda alternativa enfocada en priorizar los usuarios con mayor cantidad de escenarios es la que peor desempeño tuvo, incluso peor que sin usar planificador. La Alternativa 1 que prioriza los usuarios con menor cantidad de escenarios fue la que de menor flow time. Finalmente, la Alternativa 3 presentó tiempos peores que la Alternativa 1, con una diferencia de 180 minutos.

TABLA III
TABLA DE VALORES ARROJADOS POR LOS EXPERIMENTOS

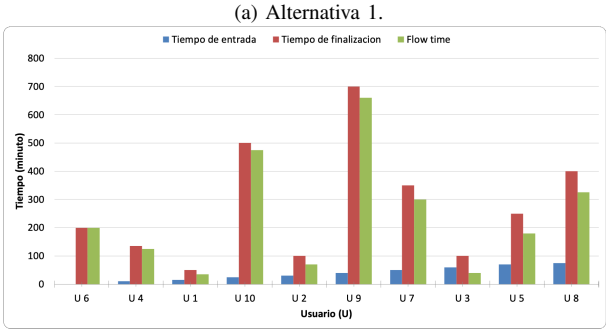
Métricas/Alternativas	1	2	3	Simugan
Flow time	1330	2729	1510	2256
Media	84	169	94	141
Desvío estándar	101	134	113	88
Limite inferior	-17 (= 0)	35	-19 (= 0)	53
Limite superior	185	304	208	228

D. Desvío Estándar del Flow Time

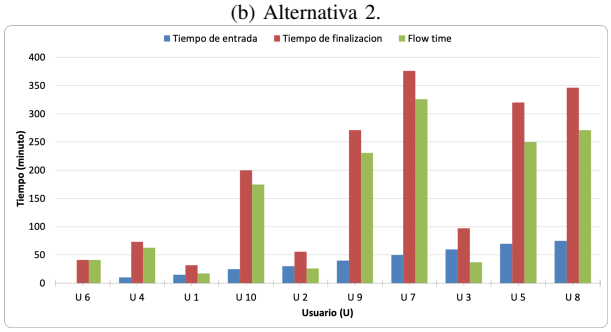
El desvío estándar del flow time es una métrica que aporta información en cuanto a la equidad en el uso del poder de cómputo de la Grid utilizado por los usuarios en relación a la cantidad de escenarios a ejecutar. El cálculo de la equidad



(a) Alternativa 1.



(b) Alternativa 2.



(c) Alternativa 3.

Fig. 2. Tiempos de ejecución usuarios en modo experimentación.

está compuesto por la medida del flow time por usuario, y un límite inferior y superior de la muestra en cuestión, donde el límite inferior y superior son iguales a la media menos y más el desvió estándar, respectivamente. El cálculo se basa en identificar cual de las tres alternativas tuvo menos valores por fuera de los límites superior e inferior.

En este sentido, la Tabla III muestra los resultados para la métricas mencionadas, donde se observa que la muestra menos dispersa es la de Simugan sin planificador con un desvío estándar de 88, luego la Alternativa 1 con un desvío estándar de 101, y finalmente alternativa 2 y 3 con valores 134 y 113 respectivamente.

En la Fig. 3 se muestra la distribución de los tiempos de los usuarios entre los límites considerados. La Fig. 3a de la Alternativa 1 muestra que sólo los usuarios 8 y 9 no están dentro de los límites aceptables. Esto se debe a que son usuarios con gran cantidad de escenarios, y dada la configuración de la Alternativa 1, fueron relegados para ejecutar usuarios con menor cantidad de escenarios. La Fig. 3b de la Alternativa 2 muestra que diferente a la Alternativa 1, el tiempo de los usuarios de experimentación se acercaron al límite superior y en el caso de los usuarios 4, 9 y 7 superándolo, mientras que los usuarios en modo simulación quedan fuera del límite inferior. En total fueron 9 los usuarios que no estuvieron dentro de los parámetros aceptables, debido a que el planificador fue priorizando usuarios con mayor cantidad de escenarios, por lo que los usuarios con poca cantidad de escenarios permanecieron mucho tiempo en cola de espera. En la Fig. 3c (Alternativa 3), a diferencia de la Alternativa 1, se observa que los usuarios con mayor cantidad de escenarios son beneficiados debido a que la configuración prioriza usuarios con escenarios en espera menores a 150, para luego priorizar a los que más cantidad de escenarios tienen en cola de espera. En el caso de esta alternativa, son 4 los usuarios que no se encuentran dentro de los límites definidos. Por último, la Fig. 3d referida a Simugan sin uso de planificador muestra que para los usuarios en modo simulación el flow time es considerablemente superior a las 3 alternativas. Asimismo, se puede observar que son 7 los usuarios por fuera de los límites y por ende los tiempos de cómputo no son equitativos.

En resumen, se puede observar que las tres alternativas con planificador presentan mejor desempeño general para los usuarios en modo simulación debido a la política de priorizar éstos versus los usuarios en modo experimentación. Sin embargo, de la Tabla IV, que muestra el resumen con la cantidad de usuarios fuera de los límites, se puede inferir que la priorización de usuarios simulación versus usuarios experimentación no siempre conlleva a un uso equitativo del poder de cómputo respecto a la cantidad de escenarios a simular.

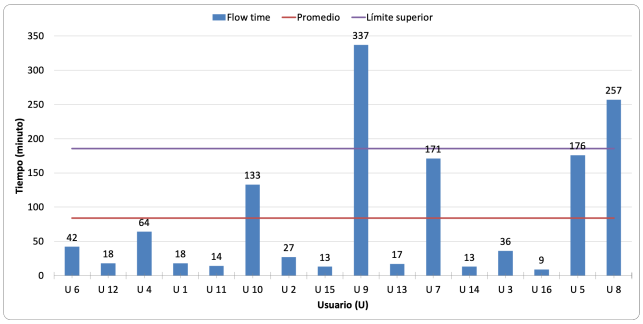
TABLA IV
TABLA RESUMEN DE LOS USUARIOS FUERA DE LOS LÍMITES CONSIDERADOS.

Alternativas	1	2	3	Simugan (sin planificador)
Usuarios fuera del los límites	2	9	4	7

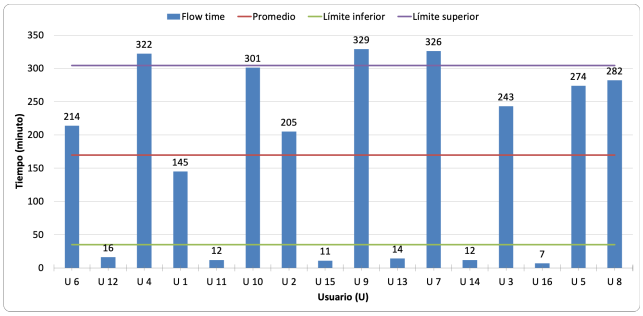
V. CONCLUSIONES

Los resultados de la experimentación muestran que el flow time para dos de las tres alternativas de priorización de escenarios propuestas mejoraron considerablemente a Simugan sin un módulo de planificación. En dichos experimentos es posible visualizar que los escenarios para usuarios en modo simulación son prioritarios y por ende los primeros en ejecutarse, para luego sí comenzar a priorizar a los usuarios que menos cantidad de escenarios tienen en cola de espera. La Alternativa 1 mejora el flow time de Simugan en un 41 %, mientras que la Alternativa 3 lo mejoró en un 33 %. Particularmente, con la Alternativa 2 se observó que al tomar los escenarios de los usuarios con mayor cantidad de escenarios, el flow time para los usuarios con menos cantidad de escenarios se incrementó considerablemente, por lo que esta alternativa no logró la mejora que se busca en el enfoque, empeorando el desempeño de Simugan en un 21 %.

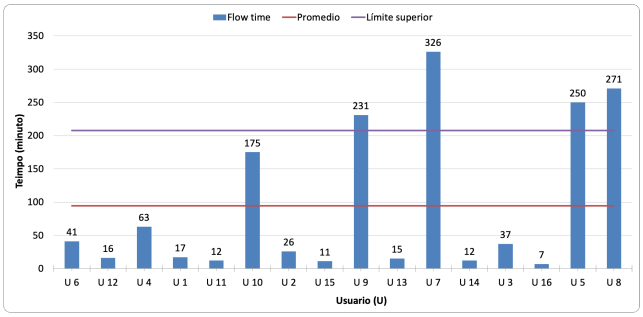
Por lo mencionado y corroborado empíricamente, se puede concluir que la introducción de un módulo de planificación mejora significativamente la experiencia de usuario en cuanto flow time. Sin embargo, una equivocada selección de la



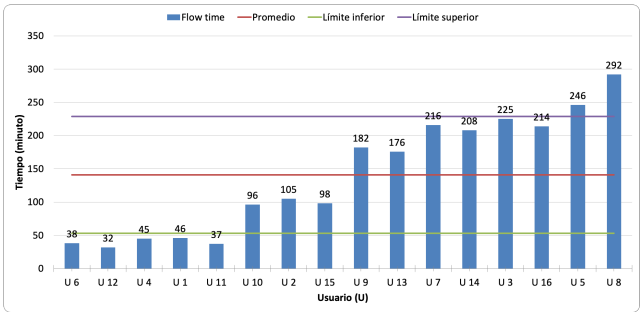
(a) Alternativa 1



(b) Alternativa 2



(c) Alternativa 3



(d) Simugan sin planificador

Fig. 3. Distribución de tiempos entre los límites fijados.

estrategia puede producir efectos nulos o hasta adversos, como en el caso de la Alternativa 2. De aquí surge la posibilidad de que la Alternativa 1, la mejor en cuanto a performance en el escenario de experimentación planteado, puede no ser la mejor en otro escenario. Se estudiarán mecanismos para que el planificador cambie de estrategia de planificación dinámicamente de acuerdo a su estado, esto es que de acuerdo al estado de las colas, el planificador pueda cambiar de alternativa de planificación, por ejemplo cambiar de la Alternativa 1 a la Alternativa 3 u otra alternativa que se estudie.

REFERENCIAS

[1] F. FAO, "The future of food and agriculture—alternative pathways to 2050," 2018.

[2] SENASA, "Servicio nacional de sanidad animal. www.senasa.gov.ar." 2020.

[3] P. Modernel, W. A. Rossing, M. Corbeels, S. Dogliotti, V. Picasso, and P. Tittonell, "Land use change and ecosystem service provision in pampas and campos grasslands of southern south america," *Environmental Research Letters*, vol. 11, no. 11, p. 113002, 2016.

[4] A. Del Prado, P. Crosson, J. E. Olesen, and C. A. Rotz, "Whole-farm models to quantify greenhouse gas emissions and their potential use for linking climate change mitigation and adaptation in temperate grassland ruminant-based farming systems," *Animal*, vol. 7, no. s2, pp. 373–385, 2013.

[5] C. A. Rotz, B. Isenberg, K. Stackhouse-Lawson, and E. Pollak, "A simulation-based approach for evaluating and comparing the environmental footprints of beef production systems," *Journal of animal science*, vol. 91, no. 11, pp. 5427–5437, 2013.

[6] A. D. Moore, R. J. Eckard, P. J. Thorburn, P. R. Grace, E. Wang, and D. Chen, "Mathematical modeling for improved greenhouse gas balances, agro-ecosystems, and policy development: lessons from the australian experience," *Wiley Interdisciplinary Reviews: Climate Change*, vol. 5, no. 6, pp. 735–752, 2014.

[7] C. F. Machado, S. Morris, J. Hodgson, M. Arroqui, and P. Mangudo, "A web-based model for simulating whole-farm beef cattle systems," *Computers and Electronics in Agriculture*, vol. 74, no. 1, pp. 129 – 136, 2010.

[8] M. Arroqui, P. Mangudo, C. Marcos, , and C. Machado, "Agile Development for a Beef-Cattle Farm Simulator," *IEEE Latin America Transactions*, vol. 7, no. 5, pp. 578–585, Sep. 2009. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5361196

[9] C. F. Machado, F. Bilotto, H. Berger, P. Mangudo, A. Arroqui, I. Balcarce, and D. MinCyT, "Evaluación de oportunidades de innovación de sistemas productivos de carne bovina con modelos de simulación," *XXIV Congr. la Asoc. Latinoam. Prod. Anim. y XL Congr. la Soc. Chil. Prod. Anim. Sochipa. AG Puerto Varas, Chile*, 2015.

[10] I. N. Stefanazzi, J. C. Burges, C. Machado, H. Berger, C. Faverín, A. J. Pordomingo, and O. N. Di Marco, "Simulation of cow-calf productivity with the use of deferred sorghum." in *Congreso Argentino de Producción Animal. 34. Joint Meeting ASAS-AAPA. 1. 2011 10 04-07, 4-7 de octubre de 2011. Mar del Plata, Buenos Aires. AR.*, 2011.

[11] H. Berger, F. Bilotto, L. W. Bell, and C. F. Machado, "Feedbase intervention in a cow-calf system in the flooding pampas of argentina: 2. estimation of the marginal value of additional feed," *Agricultural Systems*, vol. 158, pp. 68–77, 2017.

[12] F. Bilotto, P. Recavarren, R. Vibart, and C. F. Machado, "Backgrounding strategy effects on farm productivity, profitability and greenhouse gas emissions of cow-calf systems in the flooding pampas of argentina," *Agricultural Systems*, vol. 176, p. 102688, 2019.

[13] C. Fernandez Rosso, F. Bilotto, A. Lauric, G. A. De Leo, C. Torres Carbonell, M. A. Arroqui, C. G. Sørensen, and C. F. Machado, "An innovation path in argentinean cow–calf operations: Insights from participatory farm system modelling," *Systems Research and Behavioral Science*, 2020.

[14] R. V. Lopes and D. Menascé, "A taxonomy of job scheduling on distributed computing systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 12, pp. 3412–3428, 2016.

[15] R. Tyagi and S. K. Gupta, "A survey on scheduling algorithms for parallel and distributed systems," in *Silicon Photonics & High Performance Computing*, A. Mishra, A. Basu, and V. Tyagi, Eds. Singapore: Springer Singapore, 2018, pp. 51–64.

[16] B. L. Maccarthy and J. Liu, "Addressing the gap in scheduling research: a review of optimization and heuristic methods in production scheduling," *The International Journal of Production Research*, vol. 31, no. 1, pp. 59–79, 1993.

[17] K. Perakis, F. Lampathaki, K. Nikas, Y. Georgiou, O. Marko, and J. Maselyne, "Cybele–fostering precision agriculture & livestock farming through secure access to large-scale hpc enabled virtual industrial experimentation environments fostering scalable big data analytics," *Computer Networks*, vol. 168, p. 107035, 2020.

[18] J. M. Montañana, P. Marangio, and A. Hervás, "Open source framework for enabling hpc and cloud geoprocessing services," *AGRIS on-line Papers in Economics and Informatics*, vol. 10, no. 665-2021-552, pp. 61–76, 2020.
[19] F. Pan, Q. Feng, R. McGehee, B. A. Engel, D. C. Flanagan, and J. Chen, "A framework for automated and spatially-distributed modeling with the agricultural policy environmental extender (apex) model," *Environmental Modelling & Software*, p. 105147, 2021.
[20] M. Arroqui, J. R. Alvarez, H. Vazquez, C. Machado, C. Mateos, and A. Zunino, "Jasag: a gridification tool for agricultural simulation applications," *Concurrency and Computation: Practice and Experience*, vol. 27, no. 17, pp. 4716–4740, 2015.
[21] M. Longo, M. Arroqui, J. Rodriguez, C. Machado, C. Mateos, and A. Zunino, "Extending jasag with data processing techniques for speeding up agricultural simulation applications: A case study with simugan," *Information Processing in Agriculture*, vol. 3, no. 4, pp. 235–243, 2016.
[22] M. Arroqui, C. Mateos, C. F. Machado, and A. Zunino, "Restful web services improve the efficiency of data transfer of a whole-farm simulator accessed by android smartphones," *Computers and Electronics in Agriculture*, vol. 87, no. 0, pp. 14 – 18, 2012.
[23] M. Hemamalini, "Review on grid task scheduling in distributed heterogeneous environment," *International Journal of Computer Applications*, vol. 40, no. 2, pp. 24–30, 2012.
[24] E. Gamma, *Design patterns: elements of reusable object-oriented software*. Pearson Education India, 1995.
[25] E. Pacini, C. Mateos, C. G. Garino, C. Careglio, and A. Mirasso, "A bio-inspired scheduler for minimizing makespan and flowtime of computational mechanics applications on federated clouds," *Journal of Intelligent & Fuzzy Systems*, vol. 31, no. 3, pp. 1731–1743, 2016.

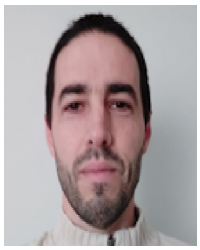
Mauricio Arroqui es Investigador Asistente del Consejo Nacional de Investigaciones Científicas y Técnicas de Argentina (CONICET) en el Instituto Superior de Ingeniería de Software de Tandil (ISISTAN) y docente de la Fac.Cs. Exactas en la Universidad Nacional del Centro de la Provincia de Buenos Aires (UNCPBA-Argentina). Obtuvo el título de Dr. en Ciencias de la Computación en el año 2014 en la UNCPBA-Argentina.

Juan Francisco Ferreira es Ingeniero de Sistemas recibido en el año 2020 en la Universidad Nacional del Centro de la Provincia de Buenos Aires (UNCPBA-Argentina).

Maximiliano Omar Roselli es Ingeniero de Sistemas recibido en el año 2020 en la Universidad Nacional del Centro de la Provincia de Buenos Aires (UNCPBA-Argentina).

Juan Rodriguez Alvarez es docente de la Fac. Cs. Exactas en la Universidad Nacional del Centro de la Provincia de Buenos Aires (UNCPBA-Argentina). Obtuvo el título de Dr. en Ciencias de la Computación en el año 2018 en la UNCPBA-Argentina.

Claudio F. Machado es profesor e investigador de la Fac. Cs. Vet., Universidad Nacional del Centro de la Provincia de Buenos Aires (UNCPBA-Argentina). Obtuvo el tÃtulo de Mg. en ProducciÃ³n Animal en el aÃ±o 1993 en la U. de Chile, Chile y en el 2004 el titulo de Dr. en Ciencia Animal en la Universidad de Massey, Nueva Zelanda.



Cristian Mateos es Investigador Independiente del Consejo Nacional de Investigaciones Científicas y Técnicas de Argentina (CONICET) en el Instituto Superior de Ingeniería de Software de Tandil (ISISTAN) y profesor de la Fac. Cs. Exactas en la Universidad Nacional del Centro de la Provincia de Buenos Aires (UNCPBA-Argentina). Obtuvo el título de Dr. en Ciencias de la Computación en el año 2008 en la UNCPBA-Argentina.

Alejandro Zunino es Investigador Principal del Consejo Nacional de Investigaciones Científicas y Técnicas de Argentina (CONICET) en el Instituto Superior de Ingeniería de Software de Tandil (ISISTAN) y profesor de la Fac. Cs. Exactas en la Universidad Nacional del Centro de la Provincia de Buenos Aires (UNCPBA-Argentina). Obtuvo el título de Dr. en Ciencias de la Computación en el año 2003 en la UNCPBA-Argentina.